Types of programming languages:

Programming languages can be categorized based on how they interact with the computer's hardware, their syntax, and their paradigms. Blow is an organized classification of common types of programming languages along with examples of each type.

Difference Between Low-level, High-level languages:

❖ Low-level languages:

➢ They are very close to the computer's hardware, they are fast and efficient, but hard for humans to read or write.

1. Machine Language:

- Written using binary (0,1).
- The only language that a computer understands directly.
- Very difficult for humans to write or debug.

♦ Example: 10110000 01100001

2. Assembly language:

- Uses abbreviated words/ symbols instead of (0,1).
- Easier than machine language but still difficult.
- Needs to be translated into machine language use an assembler.

♦ Example: MOV A, 7

ADD A, 8

❖ High level languages:

➢ Is a language that is close to human language and easy to read, write and understand. It allows programmers to write instructions using simple words, but it must be translated into machine language before the computer can execute it.

1. Python:

- Very simple and great for beginners.
- Used in AI, data analysis, web development.
- Clean syntax.
  - ♦ Example: Programming a robot - analyse data - simple web app.

2. JavaScript:

- Must have for any website.
- Controls how websites behave and interact.
- Works in every browser.
  - ♦ Example: when you click a button on a site and something happens – that's JavaScript.

3. Java:

- Strong and popular in big companies.
- Used in Android apps and enterprise systems.
- Secure, stable, and in high demand.
  - ♦ Example: Mobile banking app – school system – HR software.

4. C#:

- Created by Microsoft.
- Used for Windows apps and games.
- Very popular in game development.

- ♦ Example: Making game – creating a Windows calculator app.

5.SQL:

- Not a full programming language, but very important.
- Used to work with databases.
- Helps find and manage data.
  - ♦ Example: Show me all students with grades over 70

→ Note: Most programmers don't use low level languages today, because high level languages like (Python or Java) are much easier to learn and use.

Difference Between Compiled, Interpreted languages:

❖ Compiled Languages:

- The code is translated all at once before running the program.
- A separate executable file is created.
- You don't need the original code every time to run the program.
  - ♦ Examples: C, C++.

❖ Interpreted languages:

- The code is translated line by line at runtime (while the program is running).
- There's no standalone file; the code must be present every time.
- Slower than compiled languages.
  - ♦ Examples: Python, JavaScript.

Difference Between Programming, Scripting languages:

❖ Programming languages:

- Used to build full applications and systems.
- Needs to be compiled before running.
- Good for building big projects like desktop apps, mobile apps, or operating systems.
  - ♦ Examples: C, C++, Java, Python.

❖ Scripting languages:

- Used to automate tasks or add interactivity.
- Usually runs line by line, no compiling needed.
- Used for smaller tasks like controlling web pages or writing quick scripts.
  - ♦ Examples: Python, JavaScript.

Note: Some languages like Python or JavaScript can be both – it depends on how you use them.

❑ If you use the language to:

- Do a small, quick task.
- Like renaming files, sending emails, or printing a massage.

→ Then it's called a Scripting Language.

♦ Example:

```
Print ("hello")
```

❑ If you use the same language to:

- Build a full application.
- Like a website, game, or data analysis system.

→ Then it's used as a Programming Language.

Difference Between Open Source, Not Open-Source languages:

❖ Open-Source language:

- The source code is available to everyone.
- Anyone can view, modify, and improve the code.
- Usually free to use.
- Encourages learning and collaboration.
  - ♦ Examples: Python, VS code, MySQL.

❖ Not Open-Source language:

- The source code is hidden.
- Only the company or original developer can change it.
- Often paid software.
- You can't see how it works inside.
  - ♦ Examples: IOS, Microsoft, Windows.

Difference Between Support OOP, Not Supporting OOP languages:

❖ Support OOP language:

- Allow you can create objects that conation both data and functions.
- Help keep your code organized and reusable.
- Great for large or complex apps.
- Make it easier to modify and extend programs.
  - ♦ Examples: Python, Java, C++, C#.

❖ Not support OOP language:

- Use a step-by-step approach.
- Data and code are kept separate.
- No concept of objects.

- Less organized, harder to maintain in big projects.
    - Examples: C, Assembly.

☺